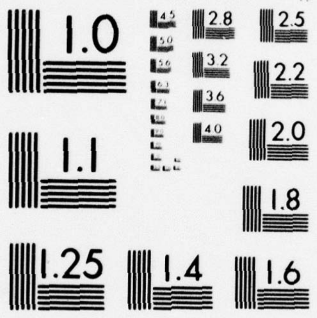
AD-A062 680　YALE UNIV　NEW HAVEN CONN DEPT OF COMPUTER SCIENCE　F/G 5/7
ON THE USE OF FRAMED KNOWLEDGE IN LANGUAGE COMPREHENSION.(U)
SEP 78　E CHARNIAK　N00014-75-C-1111
UNCLASSIFIED　RR-137　NL

1 OF 1
AD
A062680

END
DATE
FILMED
3-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

LUX ET VERITAS

ON THE USE OF FRAMED KNOWLEDGE IN LANGUAGE COMPREHENSION

Running title: ON THE USE OF FRAMED KNOWLEDGE

September 1978

Research Report #137

Eugene Charniak

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

The research described here was done at the Yale Artificial Intelligence Project and is funded in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract

15 NØØ014-75-C-1111

14 RR-137

9 Research rept,

6 ON THE USE OF FRAMED KNOWLEDGE IN LANGUAGE COMPREHENSION.

11 Sep[redacted]978

Research Report #137

10 Eugene Charniak

12 71p.

407 051

[illegible]

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER #137	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) On the Use of Framed Knowledge in Language Comprehension Running title: On the Use of Framed Knowledge		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Eugene Charniak		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-1111
9. PERFORMING ORGANIZATION NAME AND ADDRESS Yale University - Department of Computer Science Artificial Intelligence Project New Haven, Connecticut 06520		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE September 1978
		13. NUMBER OF PAGES 66
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Program Arlington, Virginia 22217		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution of this report is unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Frames, Inference, Causality, Pattern-Matching, Representation of Knowledge		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Notions like "frames", "scripts" etc. are now being used in programs to understand connected discourse. We will describe a program in this vein which understands simple stories about painting. In particular, problems of matching, read time inference, and undoing false conclusions will be stressed. The program makes heavy use of real world knowledge, and there is an extensive discussion of various issues in knowledge representation and how they affect frame representations: modularity, the need for problem solving, worldly vs control knowledge, and cleanliness.

78 12 27 021

-- OFFICIAL DISTIRUBTION LIST --

Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314	12 copies
Office of Naval Research, Information Systems Program, Code 437, Arlington, Virginia 22217	2 copies
Office of Naval Research, Code 102IP, Arlington, Virginia 22217	6 copies
Advanced Research Projects Agency, Cybernetics Technology Office, 1400 Wilson Boulevard, Arlington, Virginia 22209	3 copies
Office of Naval Research, Branch Office - Boston, 495 Summer Street, Boston, Massachusetts 02210	1 copy
Office of Naval Research, Branch Office - Chicago, 536 South Clark Street, Chicago, Illinois 60615	1 copy
Office of Naval Research, Branch Office - Pasadena, 1030 East Green Street, Pasadena, California 91106	1 copy
Mr. Steven Wong, Administrative Contracting Officer, New York Area Office, 715 Broadway - 5th Floor, New York, New York 10003	1 copy
Naval Research Laboratory, Technical Information Division, Code 2627, Washington, D.C. 20375	6 copies
Dr. A.L. Slafkosky, Scientific Advisor, Commandant of the Marine Corps, Code RD-1, Washington, D.C. 20380	1 copy

Office of Naval Research 1 copy
Code 455
Arlington, Virginia 22217

Office of Naval Research 1 copy
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center 1 copy
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E.H. Gleissner 1 copy
Naval Ship Research and Development
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper 1 copy
NAICOM/MIS Planning Board
Office of the Chief of Naval Operations
Washington, D.C. 20350

Mr. Kin B. Thompson 1 copy
Technical Director
Information Systems Division
OP-91T
Office of the Chief of Naval Operations
Washington, D.C. 20350

Advanced Research Project Agency 1 copy
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Professor Omar Wing 1 copy
Columbia University in the City of New York
Department of Electrical Engineering and
Computer Science
New York, New York 10027

Office of Naval Research 1 copy
Assistant Chief for Technology
Code 200
Arlington, Virginia 22217

ON THE USE OF FRAMED KNOWLEDGE IN LANGUAGE COMPREHENSION

Eugene Charniak

Yale University

Department of Computer Science

Running title: ON THE USE OF FRAMED KNOWLEDGE

ABSTRACT

Notions like "frames", "scripts" etc. are now being used in programs to understand connected discourse. We will describe a program in this vein which understands simple stories about painting. (Jack was painting a chair. He dipped a brush into some paint. Q: Why?) In partcular, problems of matching. read time inference, and undoing false conclusions will be stressed. The program makes heavy use of real world knowledge, and there is an extensive discussion of various issues in knowledge representation and how they affect frame representations: modularity, the need for problem solving, worldly vs control knowledge, and cleanlyness. The paper concludes with an extensive discussion of the program's shortcommings.

1. INTRODUCTION

Much recent work in artificial intelligence sees language comprehension as a process of relating language derived information to previously stored chunks of stereotyped knowledge, variously called "frames" (Minsky [1]). "scripts" (Schank and Abelson [2]). "units" (Bobrow and Winograd [3]), "depictions" (Hayes [4]), "common sense algorithms" (Rieger [5]). etc. This paper will describe a computer program, Ms. Malaprop, which is solidly within this "tradition" in that it uses a "framed" knowledge of mundane painting to answer questions about simple texts such as: "Jack was painting a chair. He dipped a brush into some paint. Q: Why?"

The paper will try to operate on two levels. On one hand I want to give a clear explanation of how Ms. Malaprop works. On the other I will emphasize certain ideas about knowledge representation. With regard to the former we have the following major points:

<u>Frame comprehension hypothesis.</u> Most systems which use frames to understand assume what I shall call the "frame comprehension hypothesis". This has it that a major part of understanding is the matching of incoming story information against the framed knowledge of what normally occurs. So, for example:

Jack was going to paint a chair. He washed it.

Q: Why? A: One should clean things one is going to paint.

(Let me note here that this, as well as all other "painting" examples are handled by the current version of Ms. Malaprop. However, the input is not English, but rather a "semantic representation" to be described later.) Ms. Malaprop understands this example because she has stored the fact that one typically cleans objects before painting them, and the second sentence will match this portion of the description of the painting act.

<u>Matching.</u> The purpose of matching is to identify the prototypical event eluded to by an occurance in the story. In general this matching can be quite complex, and involves issues of time, how close the story event is to the

prototypical event in terms of objects used, etc. So, for example:

> Jack was going to paint a chair. He dipped a brush into the paint.
>
> Q: Why? A: To get paint on it.

In order to make the match here Ms. Malaprop must identify "a brush" with the instrument of painting. Because brushes are normally used this way the identification is made and the match is successful. If we had said "He dipped a finger into the paint" this would not have been the case.

<u>Read time inference.</u> Ms. Malaprop makes many inferences while reading. It is necessary to control such inferences tightly, since in principle there is no limit on how many could be made. After we distinguish between abductive (or "explainitory") and deductive inferences, of which Ms. Malaprop only does the latter, the program recognizes the need for three kinds of deductive inferences: a) when a contradiction is noticed, b) when a frame expectation is confounded, and c) when the input imples a state or event which is important for the activities discussed in the story.

<u>Undoing wrong conclusions.</u> A system which makes inferences while reading will invariably make some false ones. Should Ms. Malaprop later discover that a particular conclusion is wrong she will correct it, as well as anything which might have been derived from it. So, for example:

> Jack finished painting. He did not clean the brush.
>
> Q: What happened? A: The brush became unabsorbent.
>
> He left the brush in the paint.
>
> Q: Did the brush become unabsorbent? A: No
>
> Q: Why not? A: Because the paint does not dry on the brush.

Note that in this example the second line matches the negation of our frame expectation that the instrument will be cleaned. Matching a statement against its negation is non standard from a logical point of view, but seems to be important in language comprehension (cf. Wilks [6]).

As mentioned above, the workings of Ms. Malaprop is only one of my concerns here. Ms. Malaprop was initially designed to exercise a knowledge representation scheme which I had previously developed [7]. As such, while the workings of the program are of interest in their own right, I tend to view the program as illustrating certain problems and theories in the domain of knowledge representation. Hence, while describing the program I will be making constant reference to a set of issues which have been of concern to me in the development of the knowledge representation used here. In particular:

Modularity. To avoid repeating the same knowledge in several different "frames" it is necessary to modularize the knowledge used. So, for example, the notion of "evaporation" comes up often in the painting domain, as well as in many other domains. If each time we needed some knowledge about evaporation (as in "why should one wash the brush afterwards", or "why close the paint can when done") we would find ourselves stating the same facts many times over in our representation. It would clearly be better (ceterus approximately paribus) to state this knowledge once, and have other frames use it when needed. Yet the traditional "frame" notion of comprehension has it that we bring in one "chunk" of knowledge in order to understand a certain stereotyped situation. We must, in other words, harmonize this idea with the opposite one of modularity.

Worldly vs. control knowledge. It is generally recognized that facts without knowledge of how to use them lead to combinatorial death. This inspired the PLANNER-like languages (like that of Hewitt [8]), where knowledge and use-knowledge were bound together in the form of programs. The trouble with this approach is that it does not allow for the most general use of the knowledge, and also makes additivity difficult. Hence Ms. Malaprop separates facts about the world from knowledge about how the facts are to be used. This separation is not as complete as that demanded by Kowalski [9] however.

Cleanliness. My goal in designing Ms. Malaprop was to make the knowledge representaion as clean as possible. This can be defended on "philosophical" grounds in that good theories are clean theories. However, at a more pragmatic level the dirtier the representation the harder it will be to write programs which use it. So the frames are precisely defined as having certain components, each of which have a certain significance, etc. One particular effort at cleanlyness is an almost complete ban on "procedural embedding" [3].

Problem solving. Probably the major objection to "frames" has been from those who see human behavior as too diverse to make stereotyped knowledge of much use. While I doubt that there is really any alternative to the sort of frames presented here, I am sufficiently impressed by this objection to want my frames to be compatible with the problem solving procedures which must surely accompany them.

Section 2 presents a review of the representation. The factual material in this section will be familiar to those who have read "Framed PAINTING" [7] although the justification of the representation in terms of the above goals is new. Section 3 outlines the probram, while 4 and 5 describe matching and read time inference in more detail. Section 6 is an extended example. Finally, Section 7 describes limitations of both the program and the representation.

2. THE FRAMED PAINTING REPRESENTATION

2.1 The Basic Representation.

As may have become apparent from the examples I have used, I am primarily concerned with "how to do it" knowledge. I am not interested in including in my representation of mundane painting the fact that "the fastest time for painting a 10 by 20 ft. wall was ..." With this in mind it is natural to see the frame as a series of instructions of the form, "first do this, then do..." So we might have:

```
(PAINTING (COMPLEX-EVENT)                    ;Anything following a ";" is a
     :                                        ;comment
     .
  GOAL:  PAINTING-GOAL                        ;Normally painting is done to
           (PAINT coats OBJECT)               ;accomplish this goal.

 EVENT:  PAINTING1 (OBJECT is clean)          ;The EVENT section describes how
         PAINTING2                            ;to paint in more detail.
          (PAPER around OBJECT)
         PAINTING3
          (LOOP
           PAINTING4                          ;Things like PAINTING3 are the
            (PAINT on INSTRUMENT)             ;names of the individual state-
           PAINTING5                          ;ments (here expressed in semi

            (INSTRUMENT in contact            ;English).
                      with OBJECT)  )
         PAINTING8                            ;The arrows, of course, indi-
          (PAINT no longer on                 ;cate time ordering.
                       INSTRUMENT)  )
```

The English-like statements will eventually be replaced by formalism, but we can already see one tendency which will continue throughout the paper. All of the statements express states to be achieved, rather than actions to be performed. This is a reflection of the fact that I want these frames to be compatible with problem solving procedures, and in most goal oriented activites we are interested in some end state rather than the exact way it is achieved. So, for example, if the OBJECT is already clean, then it is not necessary to clean it again. Hence the command is "achieve (OBJECT is clean)" rather than "(clean the OBJECT)". However, since every state in the frame is to be achieved, the "achieve", is left implicit, and only the desired state is indicated.

The frame as given is usable for both rollers and brushes and hence the neutral term INSTRUMENT has been used. (Note however that it is not good for spray guns, a problem which will not be considered here.) Clearly in one story INSTRUMENT will be one kind of object, while in a second it will be something else. That is to say that INSTRUMENT (and PAINT, and AGENT etc.) must be a variable. On the other hand there are restrictions on what sorts of things can be bound to these variables. This information will be included as follows:

```
(PAINTING (COMPLEX-EVENT)
   VARS: (AGENT (ANIMATE AGENT))        ;The agent must be animate, the
         (OBJECT (SOLID OBJECT))        ;object, solid, the paint liquid.
         (PAINT (LIQUID PAINT)
           NORMAL: (PAINT PAINT))       ;Normally the liquid used in
         (INSTRUMENT                    ;painting is paint.
          (SOLID INSTRUMENT)
          NORMAL:                       ;Normally the instrument is either
           PAINTING-BRUSH               ;a paint brush or a roller, and
             (PAINT-BRUSH INSTRUMENT)   ;absorbent. Note that two of the
           PAINTING-ROLLER              ;statements are named (e.g.,

             (ROLLER INSTRUMENT)        ;PAINTING-BRUSH). A statement is
             (ABSORBANT INSTRUMENT)  )  ;given a name if one needs to refer
                                        ;to it elsewhere.
```

(Throughout the paper we will add pieces to PAINTING. A version which includes everything mentioned in the paper is found in the appendix.) Intuitively the distinction between the absolute restricions (those which are given first) and the normal restrictions is that the former must be true of anything bound to the variable, while the latter are more like defaults. The distinction in terms of processes will be given in the section on matching (3).

Note that I have replaced the informal English with a typical "semantic representation" consisting of a predicate (e.g., PAINT-BRUSH) plus arguments (e.g., INSTRUMENT). We shall see later that PAINT-BRUSH, and indeed all predicates, are themselves frames, but this need not concern us here.

2.2 The COMES-FROM Link

PAINTING as given above explicates the action in so far as it breaks it down into simpler problems. However it does not go nearly far enough in this direction. For example, while it says to get paint on the instrument, it does not indicate how this should be done. Yet you and I know that immersing the instrument in the paint is typical, while pouring the paint over the instrument is not. Or again, the object to be painted is cleaned by washing, not by, say, dry cleaning. We will indicate such information with "COMES-FROM links" on the affected statements, as in:

```
PAINTING1 (OBJECT is clean)
  COMES-FROM: ((WASH-GOAL))
```

In effect we are saying here that the goal (OBJECT is clean) comes about by (or COMES-FROM) using the WASH frame to establish its normal goal. That is, we will have in WASH:

```
(WASH (COMPLEX-EVENT)
   VARS: ...
   GOAL: WASH-GOAL (WASH-OBJECT is clean)
  ...)
```

WASH-GOAL here is simply the name of the statement (WASH-OBJECT is clean) and as such is simply an arbitrary symbol.

I might note that having one frame know the names of statements in another will appear to many (myself included) as a blow to additivity. In section 7 I will discuss the possibility of doing away with this "feature".

To take the other example, after finishing the main loop, but before cleaning the instrument, we want to keep the paint on the instrument away from the air. This is done by leaving the instrument in the paint.

```
     PAINTING7 (NOT (ATMOSPHERE-CONTACT SOME-PAINT))
       COMES-FROM: ((LIQUID-IN4))
```

Note that SOME-PAINT here is just another variable, and is defined as being part of PAINT. In particular the program has no theory of quantification which would allow it to handle "some" in a general way.

```
   (LIQUID-IN (STATE)
    VARS: ...
    RELATIONS: ...
     (LIQUID-IN3 (LIQUID-IN OBJECT LIQUID)        ;If an object is in a liquid
       IMPLIES                                    ;then it will not be exposed
      LIQUID-IN4 (NOT (ATMOSPHERE-CONTACT OBJECT));to the atmosphere.
     ...)
```

What this states is that if we wish to keep the paint on the instrument away from the air, one way to do it is to have it (the paint on the instrument) in a liquid. Note that while this is on the right track, it is not exactly what we want, which is more or on the order of "put the instrument into the paint". We will return to this point momentarily. (The reader may also have noted that LIQUID-IN is rather different than PAINTING. Indeed it is, and as we shall see, STATE and COMPLEX-EVENT are two of five different kinds of frames in the system, but more on this later.)

A restriction placed on the representation is that anytime we have A COMES-FROM: ((B)), it must be the case that A matches B in the simple minded pattern matching sense of the term (e.g. compare PAINTING7 and LIQUID-IN4 from the last example). This insures that we have not "hidden" any information "inside" the COMES-FROM link. So the causal information linking LIQUID-IN to ATMOSPHERE-CONTACT is expressed in a separate rule. and is not expressed directly by the COMES-FROM link. The latter only indicates where the appropriate information is to be found. By using the COMES-FROM link in this way we are serving the cause of problem solving (in that we give standard methods for doing things when they are known) as well as modularity (in that the information about non-painting topics like WASH or LIQUID-IN will be expressed in frames of their own where they can be accessed by one and all).

We noted earlier that our LIQUID-IN rule was not exactly what we wanted. In fact, the LIQUID-IN rule is fine, but rather than telling us to immerse SOME-PAINT, we rather want it to suggest immersing INSTRUMENT. There is, of course, a close connection between these two possibilities, since SOME-PAINT is on the exterior of INSTRUMENT. This connection is expressed in the following rule:

```
(ATMOSPHERE-CONTACT (STATE)
 VARS: ...
 RELATIONS: ...
  ( (AND ATMOSPHERE-CONTACT3 (EXTERIOR EXT OBJ)       ;The exterior of
         ATMOSPHERE-CONTACT4 (ATMOSPHERE-CONTACT OBJ));an object is ex-
    IFF                                               ;posed to the air
   ATMOSPHERE-CONTACT5 (ATMOSPHERE-CONTACT EXT))      ;if the object is.
```

What we need to do is interpose this rule between PAINTING7 and LIQUID-IN4. That is, since PAINTING7 matches ATMOSPHERE-CONTACT5, to prevent exposure of SOME-PAINT, we only need to prevent ATMOSPHERE-CONTACT4, which we can do by immersing INSTRUMENT (via LIQUID-IN4). We express this in PAINTING as follows.

```
PAINTING7 (NOT (ATMOSPHERE-CONTACT SOME-PAINT))
 ;Until the instrument is cleaned, keep the paint on it out of the air.
  COMES-FROM: ((ATMOSPHERE-CONTACT5 ATMOSPHERE-CONTACT4)
                   ;If we keep INSTRUMENT out of the air then anything
                   ;on it will be out of the air also,
                (LIQUID-IN4 PAINTING7C (LIQUID-IN INSTRUMENT PAINT)))

                   ;so keep INSTRUMENT in the paint.
```

Here "(ATMOSPHERE-CONTACT5 ATMOSPHERE-CONTACT4)" is called an "intermediary" in that it intermediates between our initial goal PAINTING7, and the way we achieve it, LIQUID-IN4. Note that our matching restriction still holds, though in a modified form. PAINTING7 matches ATMOSPHERE-CONTACT5, while ATMOSPHERE-CONTACT4 matches LIQUID-IN4.

I have also introduced a new construction above in the form of:

```
(LIQUID-IN4 PAINTING7C (LIQUID-IN INSTRUMENT PAINT)))
```

The problem is that if we are to use the rules in LIQUID-IN we must know the bindings for the variables in LIQUID-IN. Usually Ms. Malaprop can compute these automatically because of the matching relations. In this case it is not possible, since PAINTING7 does not itself state what we actually immerse, or what we should immerse it in. To indicate these facts we give a "binder" or "frame instance" (in [10] I also called these "frame images") for the LIQUID-IN frame. This is PAINTING7C. It states that OBJECT in LIQUID-ON should be bound to INSTRUMENT and LIQUID to PAINT.

2.3 The LEADS-TO Link

Much as COMES-FROM allows us to say how a subaction is normally accomplished, the "LEADS-TO" link indicates "why" an action is performed. For example, we want to wash the instrument after finishing because otherwise it will become unabsorbent.

```
PAINTING8 (NOT (STICKY-ON SOME-PAINT INSTRUMENT))
  LEADS-TO: ((PAINT-DRY1))


(PAINT-DRY (SIMPLE-EVENT)
   VARS: ...
   EVENT: (AND                                  ;If there is paint sticking
            PAINT-DRY1 (STICKY-ON PAINT OBJECT) ;to an object, and it evap-
            PAINT-DRY2 (EVAPORATION PAINT) )    ;orates,
          CAUSES
          (AND                                  ;then the paint will become
            PAINT-DRY3 (PART-OF PAINT OBJECT)   ;part of the object, and the
            PAINT-DRY4 (NOT (ABSORBENT OBJECT)));object becomes unabsorbent.
```

(Strictly speaking, PAINTING8 does not match PAINT-DRY1, but rather its negation. This is taken care of by the program which reads in the frames. See section 2.5 .) Note that if we have further information about EVAPORATION (as Ms. Malaprop does) then we can not only understand why one washes the paint brush, but also why it is not so crucial if one leaves the brush in the paint. (No air gets at the brush, preventing EVAPORATION, which in turn prevents PAINT-DRY.)

As with COMES-FROM, the LEADS-TO pointer helps in terms of MODULARITY (the relevant information is found in two frames, EVAPORATION and PAINT-DRY) and problem solving (we are given, in particular, the information needed to do the next example).

> Jack painted a chair. He was going to throw the brush out.
>
> He did not wash it.
>
> Q: Why not? A: He was going to throw it out.

However there is another interesting aspect to the LEADS-TO link. If something goes wrong in an action, or in a story about an action, we must be able to anticipate what will occur because of the mistake. This came up in an early example where someone did not clean his brush when done. Ms. Malaprop will be able to use the LEADS-TO link to follow the consequences.

There are, of course, other ways to accomplish this same end. The SAM program (Cullingford [11]) has what are called "interference paths" within "scripts" (my complex event frames) to describe what might go wrong, and what happens as a result. Both are expressed directly in the script. From the point of view taken here, the interference approach has two problems with it. For one thing it tends to defeat modularity. What goes wrong when we do not wash the brush is exacty what goes right when we let the paint dry on the wall, or whatever it is that we paint. To express this information twice (or more) seems wasteful.

2.4 Types of Frames

During the course of the previous discussion I introduced without comment several different kinds of frames. We started with PAINTING which I call a COMPLEX-EVENT frame. But we have also seen SIMPLE-EVENT frames, and STATE frames.

A COMPLEX-EVENT frame is one like PAINTING, where the primary connective between sub-events is temporal. That is, it is of the form, first do this, then do that..." In particular, note that in PAINTING there are no causal relations between the elements of painting itself. This is yet another restriction on the representation. The idea is that any time we wish to state a cause and effect relation it should be expressed in a separate frame since it should apply to more than just the particular situation given in the COMPLEX-EVENT frame. In particular, it will be expressed in a SIMPLE-EVENT frame. SIMPLE-EVENTS consist then of a single cause and effect relation. (For reasons I no longer believe, what is intuitively a single cause and effect relation is currently expressed syntactically as two cause and effect relations. I choose to ignore this complication.) Should we wish to state that some action in a COMPLEX-EVENT is performed because of certain cause and effect relations which it enables, this can be expressed via a LEADS-TO link between the sub-event in the COMPLEX-EVENT and the SIMPLE-EVENT. What this restriction does then is to enforce a certain type of modularity on the representation.

Another type of frame which we have already come across is the STATE frame. It describes a state much as an event frame describes an action. Naturally there is no activity, but there are relations between the state and other states and actions, and it is these relations which serve to define the state.

OBJECT frames describe physical objects. They differ from, say, STATE frames in that rather than containing relations to other states they contain a description of the object along with its typical uses. An example is, the current (and incomplete) representation of PAINT-BRUSH:

```
(PAINT-BRUSH (OBJECT)
  VARS: (BRUSH)
        (BRISTLES)
        (HANDLE)
  DESCRIPTION:((SOLID BRUSH)                ;A paint brush is a solid with
               (BRISTLES BRISTLES))         :bristles and a handle. I do
               (PART-OF BRISTLES BRUSH)     ;not currently have a frame for
               (SOLID HANDLE)               ;HANDLE, so its properties (SOLID)
               (PART-OF HANDLE BRUSH))      ;are given here.
  LEADS-TO: ((PAINTING-BRUSH)) )            ;Typically used in PAINTING.
```

This should be reasonable clear save for the last line, which indicates the typical use for paint brushes. It does so by pointing to the line in PAINTING which states that normally one uses a paint brush to paint (PAINTING-BRUSH).

The remaining frame type is the ADJUNCT frame. I stated earlier that PAINTING is uncommitted as to the type of instrument. But this by itself is insufficient. After all, we do have specialized knowledge about, say, how to use a roller. If PAINTING is neutral, it cannot go there, so we need a second frame, ROLLER-PAINTING, for this information. Yet if we are to retain modularity, we do not want to have to repeat all of PAINTING in POLLER-PAINTING. It would be better simply to say that ROLLER-PAINTING is just like PAINTING except for the following differences. So the EVENT section consists of reference to PAINTING - modification pairs, each pair separated by a ":"

```
(ROLLER-PAINTING (ADJUNCT)
   VARS: (INSTRUMENT ROLLER-PAINTINGG           ;The instrument must be a

                         (ROLLER INSTRUMENT))   ;roller.
         (TRAY (ROLLER-TRAY TRAY))
 MASTER: PAINTING                               ;We are modifying PAINTING
  EVENT: (DURING PAINTING3 ROLLER-PAINTING1)    ;During the main PAINTING
           : ROLLER-PAINTING1                   ;loop make sure there is
              (FLUID-CONTAINMENT TRAY PAINT)    ;paint in the tray.

         PAINTING4 COMES-FROM :                 ;We get paint on the roller
          ((intermediaries)                     ;by rolling it in the tray
           (ROLL3 (ROLL INSTRUMENT TRAY)))

         PAINTING5 COMES-FROM : ((ROLL3))       ;We roll it along the

                                                ;object to get paint on it.
```

Ms. Malaprop can use this frame to handle examples like:

Jack was painting a wall. He rolled the roller in the tray.

Q: Has he finished yet? A: No.

Then he rolled the roller along the wall.

Q: Is this step obligatory? A: Yes.

As for how the ADJUNCT technique compares to other ways of accomplishing this sort of modularity, I point the interested reader to a discussion in [7].

Earlier I commented that predicates in the representation were themselves just frames. So, (ATMOSPHERE-CONTACT SOME-PAINT) is a reference to the ATMOSPHERE-CONTACT frame where OBJ is bound to SOME-PAINT. The idea is that given these bindings we may, if needed, infer new facts about the situation using the relations found in the frame. Or to take another example, (PAINTING JACK1 CHAIR1) is my representation for "Jack painting a chair" (tense is indicated separately). It states that we have an instance of the PAINTING frame where AGENT is bound to JACK1 and OBJECT to CHAIR1.

That each predicate is a frame is not a significant restriction on the form of the semantic representation. This is because my frames are still so loosely defined as to allow virtually anything to become one. However it does allow one to distinguish good from bad "style" in the selection of predicates for the representation of facts. For example, the reader may have noted that I have no equivalent of the ubiquitous ISA predicate. That is, rather than:

(ISA INSTRUMENT PAINT-BRUSH)

I have:

(PAINT-BRUSH INSTRUMENT)

The reason is simply that each frame is a collection of knowledge about a particular concept. We have a body of knowledge about paint brushes, hence PAINT-BRUSH is a predicate in the system. But what knowledge might we have about ISA? Well, there is inheritance of properties, but in fact this is not simply a property of ISA as is usually assumed. For example, if we know that

(LIQUID-IN INSTRUMENT PAINT) we can infer facts about INSTRUMENT just as we can with (PAINT-BRUSH INSTRUMENT) or (ISA INSTRUMENT PAINT-BRUSH). That is to say. the inheritance of properties is just a special case of the inference of properties, and as such is not a fact about ISA. But if there is no information unique to ISA there seems to be no need for an ISA frame, and hence ISA should not be a predicate in the system. (Later however we will resurrect ISA not as a predicate, but as a specialized search technique, which is, in fact what it always has been. That is, I am claiming that previous use of ISA have confused search techniques with predication.) Unfortunately, ISA is the rare case. For the most part my frames say little about which predicates are worth having.

2.5 Implementation

Barring a few minor complexities which I have left out, the representation developed here is exactly what is given to the system. It is not however what is used during the story comprehension phase. In particular there is a program which takes the list formatted versions as presented here and modifies these in several ways.

The least interesting change is from the input list format to property list structures. Internally a frame is represented as an atom with properties such as VARS, EVENT etc. A frame statement is an atom (the name of the statement) with properties such as BODY (which gives the predicate and arguments) COMES-FROM etc. The program also performs local syntactic checking (the atom COMES-FROM following a statement should be interpreted as a property of that statement, and not the name of the next statement. like PAINTING25). Arguments to predicates may be input either in positional notation (as is predominate in this paper) or in "pair" notation. However internally everything is in pair notation.

At an only slightly higher level, the frame input program adds various pointers which make it easier to examine things in frames. So, it creates a "frame index" which indexes the frame statements in a given frame according to their predicates and arguments. This is used when we are looking for a statement in a frame which matches a given statement. (This comes up particularly when we are looking for a match between what happens in the story and what we expect to happen given our framed knowledge.) We also add back pointers from individual statements to where they appear in the frame. Hence, given a COMES-FROM or LEADS-TO pointer to a statement we can find out the role the statement plays in the frame. For example, if we have a pointer from a story statement describing "washing a chair" to the "clean object" statement in PAINTING, we will be able to answer a question like "Has Jack finished" by seeing where the "clean object" statement fits in PAINTING.

Somewhat more interesting are the non-local syntactic checks (or semantic checks). If we have in PAINTING the command (STICKY-ON SOME-PAINT INSTRUMENT) Ms. Malaprop will go to the STICKY-ON frame and look at the strict requirements on the variables and see if SOME-PAINT and INSTRUMENT satisfy them. In this case she will find in STICKY-ON the following:

```
(STICKY-ON (STATE)
  VARS:  (LIQ (LIQUID LIQ))
         (OBJ (SOLID OBJ))
   ...)
```

Ms. Malaprop will then try to prove, given the information in PAINTING, the two statements, (LIQUID SOME-PAINT) and (SOLID INSTRUMENT). In this case it is fairly straightforward (both are stated explicitly in PAINTING), but in general this process will require Ms. Malaprop's inference capabilities. Only, now instead of using the story as the basic source of information, it is the frame itself which serves as the data base.

A somewhat similar situation is the checking of COMES-FROM and LEADS-TO pointers. I have mentioned that whenever we have ST1 COMES-FROM: ((ST2)) it is required that ST1 matches ST2. That is, they must both have the same predicate and non variable arguments, and the variable arguments must match ignoring different names (and in particular the absolute restrictions on both variables must be compatible). This matching has a side effect however. Consider the following case:

```
PAINTING1 (NOT (STICKY-ON DIRT OBJECT))
   COMES-FROM: ((WASH-GOAL))

WASH-GOAL (NOT (STICKY-ON BAD-STUFF OBJECT))
```

In matching PAINTING1 against WASH-GOAL we bind DIRT to BAD-STUFF and OBJECT (in PAINTING) to OBJECT (in WASH). These bindings will be recorded explicitly in PAINTING so that they need not be re-computed each time we wish to see if a particular instance of washing is plausible under the interpretation that it is being done in order to get the thing painted clean. This will give us:

```
PAINTING1 (NOT (STICKY-ON DIRT OBJECT))
   COMES-FROM: ((WASH-GOAL (WASH (OBJECT . OBJECT) (BADSTUFF . DIRT))))
```

The new item here is a frame statement which says that the instance of washing involved will have the following bindings ... . (Remember that bindings are internally specified in pair notation.)

3. THE BASIC PROCESS

Given this brief introduction to Ms. Malaprop's representation, let us now consider how the program uses it to understand stories. The fundamental idea, which we have been presupposing all along, is simply this: statements from the story are "understood" by linking them to one or more matching frame statements. So we understand the line "He got some paint on the brush" by seeing it as an instance of PAINTING4 which is the command in PAINTING telling one to achieve STICKY-ON. Once Ms. Malaprop has made this connection she can, should there be a question on the subject, use the information in the frame to answer questions about the story, such as why it was done, or how, or when. Note that in general these extra details are not filled in at read time since they are so easily obtainable from the frame. The process of linking story statements to corresponding frame statements will be called "statement integration".

Moving down one level of detail, we can view Ms. Malaprop as a combination of three components.

SET UP STORY STATEMENTS

INTEGRATE STATEMENTS INTO FRAMES

DO READ TIME INFERENCES

(There is a fourth separable section of Ms. Malaprop, DEMONSTRATE, the function in charge of inference. It does not fit into the flow chart however since it is called by all sections of the program.) Of these we have briefly described "statement integration". "Set up" is the usual sort of initial bookkeeping. "Read time inferences" are those inferences we make while reading because in the estimation of the program they are especially salient to the situation at hand. Deciding "saliency" is a very difficult problem. (See section 5)

In this section we are primarily concerned with "statement integration" but for the sake of completeness let us briefly consider what "set up" does. Take the following simple example.

> Jack was going to paint a chair with a brush. He cleaned the chair
>
> Q: Why? A: So the paint will not flake.

As we have already noted, the input to the program is not English but a semantic representation, so the first line is actually:

```
( SS-1 (INTEND JACK1
                $ST SS-2 (PAINTING JACK1 CHAIR1 PAINT-BRUSH1))
  SS-3 (PERSON JACK1)
  SS 4 (CHAIR CHAIR1)
  SS-5 (PAINT-BRUSH PAINT-BRUSH1)
       (SAME-TIME NEW-NOW (BEGIN SS-1))    )
```

Here SS-1 states that Jack intends to do SS-2, namely paint the chair. The symbols SS-1 etc. are simply names of the story statements (SS). SS 3 states that JACK1 is a person. It should be remembered that save for a few special symbols (NEW-NOW and BEGIN in the above example) all arguments to predicates are arbitrary symbols. I will, in general, end such arbitrary symbols with numbers to serve as a reminder. The last line serves to locate the events described in time. Here we simply state that "story now" (= NEW-NOW) is the point where Jack decides to do some painting.

These statements will be converted into the internal format by the same program which coverts frame statements into their internal format (see section 2.5) as frame and story statements are represented in exactly the same way. At the same time the story statements will be indexed in the data base (by the same program which for frame statements constructs the frame index). Lastly the story statements will have their arguments checked for correct types. That is if we say that (ABSORBED-BY A B) then A must be a liquid and B a solid (according to the restrictions on variables in the ABSORBED-BY frame). Again, this is done by the same programs which check frame statements for correct argument types. but now the basic source of data about the arguments will be the story, not the frame in which the statement appears. These "setting up" activities are summarized below.

	PUT LINE OF STORY INTO INTERNAL REPRESENTATION USING FRAME PARSER
Set up story statements(ss)	PUT STORY STATEMENT IN DATA BASE (DONE DURING ABOVE PROCESS)
Integrate ss's into frame	CHECK IF ARGUMENTS ARE OF THE CORRECT TYPE GIVEN THE PREDICATE
Do read time inferences	

Now we try to integrate the individual story statements into the frames. This is done via a list of "context" statements. which are simply those complex event frames (e.g. PAINTING) which have been mentioned earlier in the story. (Since the program only accepts very short stories no attempt at forgetting has been made.) Obviously for the first sentence there are no previously mentioned complex events, so no attempt at integration is made. We will. of course. add SS-2, the instantiation of the PAINTING frame. to the context list. We can then move on to line two of our story which (omitting tense information) goes as follows:

```
SS-6 (CAUSE JACK1 $ST SS-7 (NOT (STICKY-ON (OBJ . CHAIR1))))
```

This says, "Jack cause something (unspecified) to not be sticking to the chair."

(Note that since we do not know the first argument to STICKY-ON the second argument is specified by using an explicit pairing between the variable in the STICKY-ON frame (OBJ) and its binding, CHAIR1.)

This statement will be converted into internal format, asserted. and checked, as before. However this time when we move on to statement integration, there will be a previously mentioned statement on the context list. namely our painting statement. So we start looking for matching frame statements within PAINTING. The story statement we are trying to match is SS-6, the CAUSE statement, but particular knowledge of the CAUSE predication tells it to ignore the cause statement itself and concentrate on the state being caused, namely SS-7 (the "clean" statement).

In looking for a match within a frame we start by finding all statements in the frame with the same predicate (ignoring any NOT's). In the present case we will find three candidates.

```
          SS-7       (STICKY-ON  ?    CHAIR1)
      PAINTING1 (NOT (STICKY-ON DIRT  OBJECT))      ;The initial cleaning
      PAINTING4      (STICKY-ON PAINT INSTRUMENT)  ;Getting paint on initially
      PAINTING6 (NOT (STICKY-ON PAINT INSTRUMENT)) ;Cleaning instrument after
```

We must now match SS-7 against each of these to see if there is indeed a match. In general this proces is quite complex, and we will defer discussing it until the next section. But in the present case it is quite easy to rule out PAINTING1 and PAINTING6 as possibilities, since making these matches would require matching INSTRUMENT with CHAIR1, when it is already bound to PAINT-BRUSH1. Remember that (PAINTING JACK1 CHAIR1 PAINT-BRUSH1) is shorthand for:

```
(PAINTING (AGENT . JACK1) (OBJECT . CHAIR1) (INSTRUMENT . PAINT-BRUSH1))
```

Given that these bindings are no consistent with INSTRUMENT being bound to CHAIR1 there is not problem in ruling out the undesired interpretations of our story statement.

Once we have chosen a matching frame statement we add a pointer to the story statement showing where it was integrated in the context frame. In the present case we will have:

```
SS-7 (NOT (STICKY-ON (OBJ . CHAIR1))) LEADS-TO: ((PAINTING1 SS-2))
```

Here PAINTING1 is the frame statement matched with SS-7. The SS-2 serves two purposes. For one thing it indicates that we now understand SS-7 as part of the particular act of painting described in SS-2 (PAINTING JACK1 ...). Secondly, SS-2 gives the bindings of the variables which were used when we made the match between SS-7 and PAINTING1. Since the binding statement already exists, we need only give its name here.

Given this pointer Ms. Malaprop is now in position to answer all sorts of questions about SS-7. The exact mechanism is not of particular interest, but uniformly it involves following the LEADS-TO pointer back to PAINTING1 and using the formation there. To answer "why" Jack cleaned the chair we look at PAINTING1 where we see:

```
PAINTING1 (NOT (STICKY-ON DIRT OBJECT)) LEADS-TO: ((FLAKING1))
```

This states that PAINTING1 prevents flaking (as it matches the negation of one of flaking's prerequisites, the object having dirt on it). We can now use this information to explain Jack's behavior. Or again, if we are asked whether Jack has finished yet we can use the time relations in PAINTING to answer "no". Or yet again, if we should be asked if Jack could have omitted this step Ms. Malaprop will answer "Yes" on the basis of the relation between the goal of painting and PAINTING1.

With these steps included, our flow chart becomes the following:

```
                                                           Convert to internal
                                                           representation.

                                 Set up story              Assert story statements.
LOOK IN CURRENTLY ACTIVE         statements.
COMPLEX EVENT FRAMES FOR                                   Check for correct
POTENTIAL MATCHES.                                         argument types.

                                 Integrate state-
SEE IF THERE IS A MATCH          ments into frames.
GIVEN BINDINGS FROM COM-
PLEX EVENT STATEMENTS.           Do read time
                                 inferences.
IF THERE IS A GOOD MATCH
ADD POINTER FROM STORY
STATEMENT TO FRAME STM.
```

4 THE MATCHING PROCESS

One of the major problems in Artificial Intelligence is that of "recognition": that is, recognizing that some set of data is an instance of a more general case. This problem is most evident in medical diagnosis (recognizing certain complaints, test results, etc. as an instance of a certain disease), visual object recognition, and speach recognition, but it occurs in high level language comprehension as well. One way this occurs is the problem of determining which frame is relevant to a given situation. At the moment Ms. Malaprop says nothing about this difficult problem. The program depends on the relevant frames being instantiated in the input, so, if PAINTING is to be examined, we had better mention that someone is painting.

A somewhat simpler case of the recognition problem comes up when Ms. Malaprop has to recognize that a given action is an instance of some action mentioned in a complex event frame. This is what I called the "matching" problem in the last section. There we only considered a very simple case, now we will look into some of the complexities.

The matching process in Ms. Malaprop is, in fact, two separate processes. The first of these rejects potential matches on the basis of time information, while the second determines if the frame variables match the objects in the story. Let us start with time considerations.

> Jack finished painting a chair. Then he washed it.
>
> Q: Why? A: I don't know.

Ms. Malaprop's understanding of time is, for the most part, quite primitive (see section 6), but she can handle this example. First we must understand that the program does not interpret "finished painting" as meaning that every action in PAINTING has been completed, but only those actions in what is called the "center" of the frame. The center is found by first noting the action which directly leads to the goal state of the frame. The center then is simply the highest embedding LOOP (if any) around that action. (See below for the case of PAINTING.) Given this fact, Ms. Malaprop will reject all action which occur before or during the center as possible candidates for the match.

```
(PAINTING (COMPLEX-EVENT)
  VARS: ...
  GOAL: PAINTING-GOAL (EXTERIOR PAINT OBJECT)  ;The goal is coating OBJECT
          COMES-FROM: (via intermediates)
 EVENT: PAINTING1 ...

        PAINTING3
         (LOOP PAINTING4 ...                            ;THE
                PAINTING5 (CONTACT INSTRUMENT OBJECT)   ;CENTER
                ...)
        ...)
```

In a parallel fashion, actions will be rejected for being too far along in the frame. For example:

Jack was going to paint a chair. He washed the brush.

Q: Why? A: I don't know.

Although one could answer "so it is clean before using it" the important point for our purposes is that the brush cleaning is not interpreted as the cleaning recommended at the end of PAINTING. Ms. Malaprop's rule here is simply that if we are still in the pre center portion of the activity, all post-center actions are rejected.

A good example of both of these rules is the following:

Jack was going to paint. He got some newspaper.

Q: What for? A: To cover near by things.

Jack finished painting. He got some newspaper.

Q: What for? A: To clean the paint brush.

The reader might note that in the example presented in the last section (when we were integrating "Jack cleaned the chair"), the frame statement PAINTING8 (clean the instrument after) would have been rejected on time considerations before we did any variable matching. It was easier to ignore this point at that time.

But the major responsibility of action recognition falls on the variable matcher. In the last section we saw how Ms. Malaprop would not interpret cleaning the chair as failure to get paint on the painting instrument because of a mismatch between CHAIR1 and PAINT-BRUSH1. So in the simplest case an incoming story statement will match or fail to match a frame statement because its arguments are compatible (or not) with the bindings of the variables found in the frame statement. If we are told, for example, that Jack was painting a wall with a screwdriver, Ms. Malaprop would have no trouble interpreting "Jack dipped the screwdriver into the paint" as an instance of "getting paint on the instrument". If we had not been told what instrument Jack was using, such an interpretation would have been rejected on "likeliness" grounds. How such "likeliness" is implemented is the topic of the rest of this section.

The problem of likeliness comes into play when we do not know the bindings for all of the relevant variables. So consider:

Jack was going to paint a chair. He dipped a brush into some paint.

The second line is represented as Jack causing the brush to be immersed (LIQUID-IN) the paint. The match between the incomming story statement and the relevant frame statement goes as follows.

```
    SS3-2 (LIQUID-IN PAINT-BRUSH1 PAINT1)
PAINTING4 (LIQUID-IN INSTRUMENT   PAINT)
```

As opposed to the previous cases however, here INSTRUMENT and PAINT are not bound, since the statement about Jack painting did not mention either, but only the AGENT (JACK1) and the OBJECT (CHAIR1).

The reason "likeliness" must come into play is that the program must not be "fooled" by superficially similar situations such as:

Jack was going to do some painting. He washed his hands. (He is not washing the OBJECT. That is, he is not going to paint his hands.)

Jack was going to do some painting. He dipped a pencil in the paint. (He is not getting paint on INSTRUMENT.)

To prevent such mismatches Ms. Malaprop places "likeliness" restrictions on what will be allowed to match frame variables. Most important here are the normal conditions on variables. If the story object satisfies a normal condition on the variable (with some exceptions to be mentioned later) then it will be allowed to match the variable. In the case of PAINT-BRUSH1 in the example above, we will be trying to match INSTRUMENT, which has the following variable entry:

```
(INSTRUMENT (SOLID INSTRUMENT)
            NORMAL: PAINTING-BRUSH  (PAINT-BRUSH INSTRUMENT)
                    PAINTING-ROLLER (ROLLER INSTRUMENT)
                    PAINTING-ABSORB (ABSORBANT INSTRUMENT))
```

PAINT-BRUSH1 will, of couse, satisfy PAINTING-BRUSH, and hence will be considered a GOOD match for instrument. (As we shall see, the matcher rates matches either BAD, POSSIBLE, SATISFACTORY, or GOOD. If the variable is already bound to the object it is a GOOD match, if it is bound to something else it is BAD.) If none of the normal conditions are matched it is a BAD match.

This is the basic idea, but there further complications. For one thing, Ms. Malaprop distinguishes between those normal conditions which specify what kind of object we want (a "roller" or a "brush"), and those which only describe desirable properties which the object should have ("absorbency"). If the only satisfied normal conditions are of this latter type then the match is marked POSSIBLE. The effect of this is to not make the match initially but to wait for further evidence in the form of a second attempt to make the same identification. For example:

> Jack was going to paint a chair. He dipped a sponge into some paint.
>
> Q: Why? A: I don't know.
>
> Then he drew the sponge across the chair.
>
> Q: Why did John dip the sponge into the paint.
>
> A: To get paint on it.

After line two we have made one attempt to match SPONGE1 against INSTRUMENT. This has produced a POSSIBLE rating for the match of line two aginst the framed knowledge that dipping is the standard way to get paint on the instrument. However we do not act on this and hence Ms. Malaprop cannot answer the first question. When the next line comes in we make a second attempt at the match, and this time it goes through, allowing the program to answer the question it failed to answer one line earlier.

I will skip the other complications except to mention that if there are no normal conditions on a variable , but the absolute restrictions are matched then the match is graded SATISFACTORY. As described the matching procedure goes as illustrated below.

```
                 Is VAR already bound?                             MATCH RATEING

                no                  yes

    Have we already made                    To STORY-OBJECT?   no      BAD
    a POSSIBLE match
    to STORY-OBJECT?                            yes                    GOOD

        no       yes                                                   GOOD

          Does STORY-OBJECT satisfy all of the
          necessary conditions on VAR?                     no          BAD

                          yes

          Are there any normal conditions on VAR?          no          SATISFACTORY

                          yes

          Does STORY-OBJECT satisfy any of them?           no          BAD

                          yes

          Are any of these conditions on the kind
          of object which is normally used                 no          POSSIBLE

                          yes                                          GOOD
```

Next we update our flowchart to include this sections additions.

```
                                                        Convert to internal
                                                        representation.
 Find potential matches.
                                 Set up story           Assert story statements.
 ELIMINATE THOSE WHICH DO        statements.
 NOT FIT DUE TO TIME CON-                               Check for correct
 STRAINTS.                                              argument types.
                                 Integrate state-
 SEE IF THERE IS A GOOD,         ments into frames.
 SATISFACTORY, OR POSSIBLE
 MATCH ACCORDING TO ABOVE.       Do read time
                                 inferences.
 IF ONLY POSSIBLE, NOTE,
 BUT POSTPONE INTEGRATION,
 ELSE ADD LEADS-TO POINTER
 TO STORY STATMENT.
```

5. READ TIME INFERENCES

5.1 Abductive and deductive inferences

Most programs which attempt to understand text make inferences about the text while the program is "reading" rather than always waiting until a question is asked. Often (indeed most of the time) these inferences are not in any sense logically entailed by the story, so the machine can be though of as "jumping to conclusions" about what is being described in the text. While it is probably not feasible to omit such inferences (see Charniak [12]), their use is problematic since one cannot make all possible inferences. The question then becomes which ones should the program make?

Roughly speaking we can clasify read time inferences into two catagories, "abductive" and "deductive" inferences. So, for example, if we read that Prof. X submits a paper to a low status conference taking place in Tahiti, we might make the abductive (or "explanatory") inference that he is primarily interesting in a free, or at least tax deductible, trip. We are explaining his actions in terms of certain motivational hypotheses. We might also make some deductive inferences such as "he mailed a copy of the paper to the program chairman".

The frame recognition problem discussed in section 4 is one variety of abductive inference, and as I noted there, how such inferences are made is a tough problem and one Ms. Malaprop does not tackle. Hence in this section we will be concerned with deductive inferences.

While the problem with abductive inferences is how to make them, the problem with deductive inferences is which ones to make. It is all too easy, given, say, the fact that Jack is a person to infer that he has a heart, two arms, etc. But why bother? To answer this question Ms. Malaprop includes a rudimentary theory of deductive inferences. In particular she recognizes three types: "consistency inferences", "unexpected situations inferences", and "needed fact inferences".

The first of these cares for the case where a new fact contradicts an old one and a decision must be made as to which one is correct. This happens, of course, because a previous read time inference jumped to an incorrect conclusion. The second occurs when something unexpected (according to the active frames) takes place and we want to find out what the consequents will be. The final one is more complex. The general idea is that the proper representation for a situation will depend on the context in which the situation is embedded. Sometimes this is handled as part of parsing. In Ms. Malaprop this is considered part of the read time inference procedure.

5.2 Consistency and Unexpected Situation Inferences

Consistency inferences are needed any time we put something in the data base which is the direct negation of something we alreay know. If the old fact has been used to make any inferences, new statements negating these inferences will be added to the data base causing the process to be repeated. In order to do this Ms. Malaprop keeps what might be called "simple minded" data dependencies. That is, any time one fact is used to infer another, this is recorded on both facts. I call these "simple minded" data dependences to distinguish it from the full fleged data dependencies of Doyle [13] which are also able to handle cases where one fact was inferred from the absence of a second fact. The Doyle program also has the ability, given a contradiction, to use the data dependencies to determine the initial assumptions which caused the inference procedure to go wrong in the first place. Ms. Malaprop keeps track of the necessary information to do this, but makes no use of it. In the next section we will give considerable detail on an example which makes use of consistency inferences.

The second kind of deductive inferences mentioned above, "unexpected situation" inferences figures out what will happen when one of the frame expectations is confounded. This is done on the grounds that it is these situations which are most likely to be important in the story. So for example, we might be told:

Jack was painting a chair. He had too much paint on the brush.

Here the second line is understood by looking for a constraint which states that one should remain under a threshhold on the volume of paint on the brush. If this is found we state that his was not done in the current case. That is, the second line is interpreted as:

```
(NOT (THRESHHOLD-UNDER VOL-PAINT-ON-BRUSH1))
```

which then matches the frame statement:

```
PAINTING7 (THRESHHOLD-UNDER SOME-PAINT-VOL)  LEADS-TO: ((DRIP2))
```

Ms. Malaprop will infer the consequences of the disobedience by following the LEADS-TO pointer attached to the matched frame statement. Ms. Malaprop normally understands PAINTING7 as stateing that it is designed to frustrate the DRIP process by insuring that one of its prerequisites is not met. However, in this case it is, so Ms. Malaprop asks the inference maker (see section 5.4) to prove that the drip will occur.

```
(DRIP (SIMPLE-EVENT)
  VARS: (LIQ (LIQUID LIQ))
        (OBJ (SOLID OBJ))
        (LIQ-VOL (VOLUME LIQ-VOL LIQ))
        (LIQ-PART (AND (LIQUID LIQ-PART)
                       (PART-OF LIQ-PART LIQ)))
 EVENT: (AND DRIP1 (STICKY-ON LIQ OBJ)          ;If there is liquid
             DRIP2                               ;sticking to an object,
              (NOT (THRESHHOLD-UNDER LIQ-VOL)))  ;and its volume exceeds
          CAUSES                                 ;a threshhold, this can
        DRIP3 (NOT (STICKY-ON LIQ-PART OBJ)))    ;cause some of the liquid
                                                 ;to separate off.
```

(In fact, DRIP2 is really represented as "THRESHHOLD-OVER" rather than "NOT THRESHHOLD-UNDER" but this would add non-essential complications to the explanation.) Proving that there will be a drip inturn involves proving that the other prerequisite is also true (DRIP1 (STICKY-ON LIQ OBJ)). This will be easy since part of story input will be the definition of VOL-PAINT-ON-BRUSH1, which is:

```
(STICKY-ON PAINT-ON-BRUSH1 PAINT-BRUSH1)

(VOLUME VOL-PAINT-ON-BRUSH1 PAINT-ON-BRUSH1)
```

Hence Ms. Malaprop has inferred that there will be a drip.

5.3 Needed Fact Inferences

The last of the three resultant read time inferences, "needed fact inferences". To see the necessity of this, we should first note that often one can make a large number of inferences from a single story statement. That X is HUMAN allows one to infer many structural properties of X. That some object is LIQUID-IN some liquid allows one to infer STICKY-ON and NOT ATMOSPHERE-CONTACT relations, as well as a host of others (SURROUND, DISPLACEMENT etc.) not currently implemented. Which of these facts is important will differ from case to case. One scheme would, of course, simply to make all of the inferences. My guess is that this is in general impractical, but it is instructive to visualize how it would work. In the case at hand, the result of this will be to replace a complex predicate, LIQUID-IN, with more basic ones (more basic in the sense that while LIQUID-IN imples STICKY-ON, the reverse is not true). That is to say, we could replace LIQUID-IN with a standard representation consisting of a host of simpler predicates. In such a case we would not need to identify which fact is needed in a given case, since we would have them all. Whether or not such a standard representation is logically possible, it seems unlikely to be heuristically feasible. Hence we need some way to identify the needed fact in a given situation.

To take a particular example, early in the event of painting, should we learn of (LIQUID-IN INSTRUMENT PAINT) it would be important to realize that this will cause (STICKY-ON SOME-PAINT INSTRUMENT) for it is this relation which is crutial to our ability to paint. Later however, when the painting proper is done, we might be more interested in knowing that because of LIQUID-ON the air cannot dry out the brush. Ms. Malaprop handles this problem by relying on the frames themselves to indicate the fact we need to infer from a particular predicate at any time. This is done in the following way. The command in PAINTING telling us to get paint on the brush has the following format:

```
PAINTING4 (STICKY-ON SOME-PAINT INSTRUMENT)
  COMES-FROM: ((LIQUID-IN2 PAINTING4C (LIQUID-IN INSTRUMENT PAINT)))
```

The COMES-FROM link here says that we achieve PAINTING4 through the following fact:

```
    LIQUID-IN1 (LIQUID-IN OBJ LIQ)
      CAUSES
    LIQUID-IN2 (STICKY-ON SOME-LIQ OBJ)
```

More precisely, the COMES-FROM pointer says that PAINTING4 matches LIQUID-IN2, where the variables in the LIQUID-IN frame are to be bound according to PAINTING4C. The fact that we achieve PAINTING4 through the rule taking one from LIQUID-IN1 to LIQUID-IN2 is implicit in the COMES-FROM formalism.

Now, to take a second, but related, example, suppose we are at the begining of a story and we are told that Jack dipped the brush into the paint. As mentioned earlier, the crucial point of "dip" is "cause to be LIQUID-IN" and at some point during the processing of this sentence we will try to integrated (LIQUID-IN PAINT-BRUSH1 PAINT1). Assuming for the moment that this will be matched with PAINTING4C, then the fact that PAINTING4C hanges off PAINTING4 via a COMES-FROM pointer will indicate to Ms. Malaprop that PAINTING4 is the needed fact, given that we have seen LIQUID-IN in the current situation.

In fact, there will be initially more that one potential match in PAINTING for our LIQUID-IN story statement. The other will be found off PAINTING8.

```
  PAINTING7 (NOT (ATMOSPHERE-CONTACT SOME-PAINT))
    COMES-FROM: ((ATMOSPHERE-CONTACT5 ATMOSPHERE-CONTACT4)
                 (LIQUID-IN4 PAINTING7C (LIQUID-IN INSTRUMENT PAINT)))
```

PAINTING7 is the command which states that after the main painting loop, but before washing, the paint on the instrument should be kept out of contact with the air. The way one normally does this is to leave the instrument in the paint. In the above situation where we have just started painting, Ms. Malaprop will be able to decide in favor of PAINTING4C over PAINTING8C as the appropriate match on the basis of time considerations. However, should we later be told that Jack had finished painting, and then put the brush in the paint, we would now match the LIQUID-IN story statement against PAINTING8C, again on the grounds of appropritate time relations. Furthermore, since PAINTING8C is hanging off of a NOT ATMOSPHERE-CONTACT statement, this latter will now be taken as the needed fact given the LIQUID-IN statement.

With the addition of the various types of read time inferences, our "flow chart" for Ms. Malaprop appears below.

		Convert to internal representation.
Find potential matches	Set up story statements.	Assert story statements.
Eliminate some using time constraints.		Check for correct argument types.
See if the match is reasonable.	Integrate statements into frames.	
		IF INPUT CONTRADICTS PREVIOUS INFERENCE, UPDATE IT AND ANYTHING INFERED FROM IT.
Add pointer to story statement indicateing match.	Do read time inferences.	IF IT CONFOUNDS FRAME EXPECTATIONS, INFER CONSEQUENCES.
		IF FRAME INDICATES A NEEDED FACT, INFER IT.

5.4 The Inference Mechanism

So far I have said little about how inference making is actually done in Ms. Malaprop. On the other hand, the inference mechanism is of little interest in its own right, except to the degree that it relates to the higer level goals of the program, such as restricted representation formalism or separation of facts from use. So we will make a quick pass emphasizing how its abilities relate to the major representational goals.

Given some fact which Ms. Malaprop would like inferred, the inference maker will call on the following abilities, in roughly the following order:

Retrieval from story data base.

Retrieval from active COMPLEX-EVENT and ADJUNCT frames.

Use of ISA hierarchy on OBJECT frames.

Use of rules in STATE or SIMPLE-EVENT frames.

Use of LISP programs.

The first of these is properly speaking not an inference technique at all. The second was mentioned earlier (section 2.3) when we noted how restricting COMPLEX-EVENT frames to the representation of desired states of affairs simplifies the program needed to infer things from their presence in a COMPLEX-EVENT frame. These two techniques are considered "low cost" techniques, and are the only ones used when the inferencer is told to use LOW effort. The rest are used for NORMAL effort. (HIGH effort allows Ms. Malaprop to assume things to be true in order to explain certain facts, hence giving Ms. Malaprop a limited ability to do "explanatory read time inferences" (see section 4.1). For example this is used in:

Jack was going to paint the chair green. He got some blue and yellow paint.

Q: Why did Jack get the yellow paint? A: To mix with the blue pain t.

Here the program assumed a "mixing" action in order to explain the acquisition of the different colored paints. I am ignoring this facility however because it is quite ad hoc.)

Let us assume that Ms. Malaprop is trying to prove (PHYS-OB PAINT-BRUSH1). Assuming the first two techniqes do not work, she looks at the frame for the predicate involved (PHYS-OB) for the section labeled IF-NEEDED. This section can give three kinds of recommendations, one corresponding to each of the last three inference methods. In the present case we will find:

```
(PHYS-OB (STATE)
  VARS: (OB)
  RELATIONS: ( PHYS-OB1 (PHYS-OB OB)
                IFF
               (OR PHYS-OB2 (SOLID OB)
                   PHYS-OB3 (LIQUID OB)
                   PHYS-OB4 (GAS OB))
  IF-NEEDED: ((INFER-FROM PHYS-OB1)))
```

The IF-NEEDED section says to use the above rule to prove that something is a physical object. Like the other pointers, (e.g. LEADS-TO) the statement PHYS-OB1 is constrained to match the statement composed of the frame name plus its variables. By using this pointer we separate the fact from the information on how to use it.

Having located the relevant rule, the inferencer will try to prove the disjunction, and in particular will try to prove (SOLID PAINT-BRUSH1). Again assuming this is not in the data base, the inferencer will consult the SOLID frame, there to find:

```
IF-NEEDED: ((ISA-LINK SOL))
```

This states that to prove something is a solid, use the ISA inference method. What this means is that the program will look at the OBJECT frames which describe SOL (in this case SOL will be bound to PAINT-BRUSH1) for a statement matching that which we wish to prove. In this case it will succeed in the

PAINT-BRUSH frame (see the frame as given in section 2.4). If it had not, then it would have used OBJECT frames found in the description section of PAINT-BRUSH to continue the search. (This corresponds to the transitivity of ISA.)

The banishment of the ISA predicate however has created a problem for the ISA search technique. Typically we will have stored several OBJECT frames describing the same story object. So CHAIR1 might be described as a CHAIR, and also as a (piece of) FURNITURE. (SOLID, however is a STATE in Ms. Malaprop.) This means that the ISA technique might first search FURNITURE, and if that failed, search CHAIR, and then because CHAIR mentions that chairs are furniture, search FURNITURE all over again. What we need is a pointer to the most restricted object type we know about a given story object, in this case (CHAIR CHAIR1). The problem for the frame representation is the status of this pointer (which for the sake of old times we can call the ISA pointer). It should not be a predicate for the reasons given in section 2.5, but what it should "be" I don't know. At any rate, it does not exist in Ms. Malaprop, but eventually something like this is going to be needed.

The last of the three techniques is the use of a LISP program to decide the issue. As I have indicated, my approach is to limit the use of arbitrary programs. At the present time the use of LISP code for inference occurs in three situations. The first is the ad hoc uses (nobody's perfect), but only occurs two places in the code. (One will be removed with the addition of a fourth IF-NEEDED type, PRESUMABLY (McDermott [14]), which states that something is to be presumed true unless we can show otherwise. The other has to do with the difficulity of representing information about how colors mix. I escaped this problem by writing a LISP program which given two colors, and a desired outcome, returns T if the two colors when mixed will give the third.)

Secondly, LISP code is used for question answering routines. I have not gone into the method by which Ms. Malaprop actually answers questions because this was not a major concern of my research. What in fact happens is that a statement is constructed such as (WHY JACK1 SS2-2 X) ("X is why JACK1 did SS2-2), and Ms. Malaprop tries to "prove it". In the course of this X will be bound. To prove it Ms. Malaprop consults the WHY "frame" for an IF-NEEDED method, and finds the question answering program. This is obviously ad hoc, and should eventually be replaced by a more sophisticated question answering system, such as described by Lehnert [15].

Finally there is a class of LISP inference rules which are at least defensible. For example, Ms. Malaprop in trying to prove that an action is obligatory in a given situation tries to show that it is needed in order to accomplish the actor's goal. To do this we need to prove (GOAL G ACT) ("G is the goal of action ACT"). To do this in turn we need a way to retrieve the GOAL section of the frame ACT, and this is most naturally done with a LISP program. (David Barstow has pointed out (personal communication) that this last use of LISP could also be eliminated by making such predicates primitives, in the same way as NOT, AND, etc are primitive. In the long run this is probably the correct thing to do, but while everything is in flux, it is easier to add new frames with LISP program IF-NEEDED methods.)

6 AN EXAMPLE IN DETAIL

To get a better idea of how this all fits together, let us now take one example and go through it in detail. The example is:

> Jack finished painting. He did not clean the paint brush.
>
> Q: What happened? A: The brush became unabsorbant.
>
> He put the brush into the paint.
>
> Q: Did the paint dry on the brush? A: No.
>
> Q: Why did Jack put the brush in the paint?
>
> A: To prevent the paint from drying.

The actual input for this is:

```
((SS-1 (PAINTING JACK1)                            ;Jack's painting activity
  SS-2 (PERSON JACK1)
       (BEFORE (END SS-1) NEW-NOW))                ;is finished.

 (     (SAME-TIME OLD-NOW (BEGIN SS-3))            ;Now he
  SS-3 (NOT                                        ;fails to
        (CAUSE JACK1                               ;cause
               $ST (NOT SS-4 (STICKY-ON 1          ;the removal of
                               SOME-PAINT1         ;the paint from
                               PAINT-BRUSH1))))    ;the brush.
  SS-5 (PAINT-BRUSH PAINT-BRUSH1)
  SS-6 (PAINT PAINT1)
  SS-7 (PAINT SOME-PAINT1)
  SS-8 (PART-OF SOME-PAINT1 PAINT1)
       (SAME-TIME NEW-NOW (BEGIN SS-4)) )

 (ANSWER (X)(WHAT-HAPPENED SS-4 X))                ;Q:  What happened due
                                                   ;to SS-4?
 (       (SAME-TIME OLD-NOW (BEGIN SS-10))         ;Now
         (BEFORE (END SS-1) OLD-NOW)
  SS-9 (CAUSE                                      ;Jack causes
         JACK1
         $ST SS-10 (LIQUID-IN PAINT-BRUSH1         ;the paint brush to be
                              PAINT1))             ;in the paint.
         (SAME-TIME NEW-NOW (BEGIN SS-10)) )

 (ANSWER (X)(WHY JACK1 SS-9 X)) )                  ;Q:  Why did he do it?
```

The first line simply sets up an instance of the painting frame with JACK1 as the agent. The time information states that the action is finished. (Remember that Ms. Malaprop only assumes that the center of the action is finished.) Since prior to this line there were no complex events, there is noting further to be done with this line, except to note that PAINTING is itself a complex event, and so is put on the context list.

Coming to line two, we try to integrate it into PAINTING. Here there is a bit of ad hocery in that (NOT (CAUSE (NOT X))) is translated into X. That is, in the present case the second line becomes, in effect "there was paint on the brush". Needless to say, this does not capture the full import of the original in that it does not suggest that the paint will remain there although the original does so suggest. Even worse, Ms. Malaprop treats the revised version as if it implies the paint will stay there. The two mistakes cancel each other out, but this is clearly a bad portion of the program. It is all comparatively kludge free from here on.

So the incoming line is treated as if it read:

```
(STICKY-ON SOME-PAINT1 PAINT-BRUSH1)
```

This will be matched against the statments in PAINTING, and simply on the basis of the predicate, three will be considered: the command to clean the object, the command to get paint on the instrument, and the command to get paint off again. The first will be eliminated on time grounds but it would have been eliminated anyway since SOME-PAINT1 does not match the variable DIRT, and PAINT-BRUSH1 is a bad match for OBJECT. The second (get paint on the brush) is also eliminated on time grounds (given that Jack has finished, only statements after the center will be considered). This leaves the third. It will match because SOME-PAINT1 is a GOOD match for PAINT, and PAINT-BRUSH1 is a GOOD match for INSTRUMENT (they both satisfy OBJECT normal conditions on the variables). The net effect will be to bind these two PAINTING variables to the story objects, and to add the following to SS-4:

```
SS-4 (STICKY-ON SOME-PAINT1 PAINT-BRUSH1)  LEADS-TO: ((PAINTING8 SS-1))
```

We then move on to do read time inference. Here we note that SS-4 is in fact the negative of PAINTING8, and hence represents something we were not expecting. So Ms. Malaprop tries to find out what will happen. She does this by following the LEADS-TO pointer on PAINTING8.

```
PAINTING8 (NOT (STICKY-ON PAINT INSTRUMENT))
  LEADS-TO:  ((PAINT-DRY1 PAINT-DRY4)
              (PAINTING-ABSORB))
```

(In section 2 I gave PAINTING8 without the intermediary, but this was only to simplify discussion.) What the LEADS-TO says is that PAINTING8 leads to the negation of PAINT-DRY1 which via the rule of paint drying can cause PAINT-DRY4, which inturn matches the negation of PAINTING-ABSORB (the requirement that the instrument be absorbent). Or to put this more succinctly, if we clean the instrument we prevent the paint from drying on it which would cause loss of instrument absorbency.

```
EVENT: (AND                                      ;If there is paint sticking
         PAINT-DRY1 (STICKY-ON PAINT OBJECT)     ;to an object, and it evap-
         PAINT-DRY2 (EVAPORATION PAINT) )        ;orates,
       CAUSES
       (AND                                      ;then the paint will become
         PAINT-DRY3 (PART-OF PAINT OBJECT)       ;part of the object, and the
         PAINT-DRY4 (NOT (ABSORBANT OBJECT)));object becomes unabsorbent.
```

Ms. Malaprop wants to use this structure to understand what will occur. What she does is to try to prove that (NOT PAINTING-ABSORB) will occur. The presence of the intermediaries indicate that she cannot do this directly, but must first prove that the intermediary relation holds, which involves proving that PAINT-DRY4 will occur. To do this she needs to prove PAINT-DRY1 and PAINT-DRY2. She already has PAINT-DRY1, that is what started this in the first place. She now tries to prove that the paint will evaporate. This is not in the data base, nor is it to be found in PAINTING itself. So Ms. Malaprop goes to the EVAPORATION frame looking for advice. There she finds.

```
(EVAPORATION (SIMPLE-EVENT)
  VARS: (LIQ (LIQUID LIQ))
  EVENT: EVAP1 (ATMOSPHERE-CONTACT LIQ)            ;Being in contact with

          CAUSES                                   ;the air causes a

         EVAP2 (EVAPORATION LIQ)                   ;liquid to evaporate.
  IF-NEEDED: ((INFER-FROM EVAP2)))
```

The IF-NEEDED advice is to prove EVAPORATION by proving ATMOSPHERE-CONTACT. This is not in the data base either, but in ATMOSPHERE-CONTACT we find:

```
(ATMOSPHERE-CONTACT (STATE)
  VARS: (OBJ (PHYS-OB OBJ))
        (EXT (PHYS-OB EXT))
  RELATIONS:
    ( ATMOSPHERE-CONTACT1 (PHYS-OB OBJ)                ;Things are usu-

       IMPLIES SOMETIMES                               ;ally exposed to
      ATMOSPHERE-CONTACT2 (ATMOSPHERE-CONTACT OBJ))    ;the atmosphere.
    ( (AND ATMOSPHERE-CONTACT3 (EXTERIOR EXT OBJ)      ;The exterior of
           ATMOSPHERE-CONTACT4 (ATMOSPHERE-CONTACT OBJ));something is in
       IFF                                             ;contact with the
      ATMOSPHERE-CONTACT5 (ATMOSPHERE-CONTACT EXT))    ;atmosphere if the
  IF-NEEDED: ((INFER-FROM ATMOSPHERE-CONTACT2))        ;object is.
```

The IF-NEEDED tells us to use the first rule, so we do so. This rule simply requires that OBJ is a physical object, which of course, SOME-PAINT1 is. (In fact, given that OBJ is defined as a PHYS-OB, ATMOSPHERE-CONTACT1 will always be true. The reason it is there is because the representation has no facilities for expressing facts except as implications, something which must eventually be corrected.) However, note that this rule is marked SOMETIMES. The procedural correlate of this is that before we use the rule, the inferencer first tries to prove (using LOW effort) that PAINT-BRUSH1 is not in contact with the atmosphere. This will fail, so we then use rule 1 and infer (ATMOSPHERE-CONTACT SOME-PAINT1). This in turn is used by EVAP1 to infer (EVAPORATION SOME-PAINT1), which in turn is used by PAINT-DRY2, an enables us to prove, as we set out to do, PAINT-DRY4. In the course of doing this we will keep track in the data base how we inferred each fact using COMES-FROM pointers to show the rule used for inferring a certain fact, and LEADS-TO to show what was deduced from the fact. The net result of this will be the appearance of the following statements in the data base:

```
SS-4 (STICKY-ON SOME-PAINT1 PAINT-BRUSH1) ;This leads to both the negation
  LEADS-TO: (AND ((PAINTING6 SS-1))       ;of the command in PAINTING and
                 ((PAINT-DRY1 SD-1)))     ;a pre-requisite of PAINT-DRY.

SD-1 (PAINT-DRY SOME-PAINT1 PAINT-BRUSH1) ;This statement serves to bind
                                          ;the variables in PAINT-DRY.

SD-2 (EVAPORATION SOME-PAINT1)            ;SD-2 is inferred from the rule
  COMES-FROM: ((EVAP2 SD-2))              ;which includes EVAP2 with SD-2
  LEADS-TO: ((PAINT-DRY2 SD-1))           ;as binder.

SD-3 (ATMOSPHERE-CONTACT SOME-PAINT1)
  COMES-FROM: ((ATMOSPHERE-CONTACT2 SD-3))
  LEADS-TO: ((EVAP1 SD-2))

SD-4 (NOT (ABSORBANT PAINT-BRUSH1))
  COMES-FROM: ((PAINT-DRY4 DS-1))
  LEADS-TO: ((PAINTING-ABSORB SS-1))
```

Now we get the question, "What happened". (Note that this is given to the program in the form "what happened due to not cleaning the brush".) Ms. Malaprop simply needs to follow the LEADS-TO chain from SS-1 to SD-4 to get the answer.

We first attempt to integrate SS-9 which is the statement that Jack caused SS-10 (LIQUID-IN PAINT-BRUSH1 PAINT1). So we integrate SS-10, and find two matches in PAINTING, one concerned with getting paint on the brush, the other with keeping air away from it. By time considerations we choose the second, PAINTING7C which appears in:

```
PAINTING7 (NOT (ATMOSPHERE-CONTACT SOME-PAINT))
  COMES-FROM: ((ATMOSPHERE-CONTACT5 ATMOSPHERE-CONTACT4)
               (LIQUID-IN4 PAINTING7C (LIQUID-IN INSTRUMENT PAINT)))
```

This is one of those cases which call for a needed fact inference, in this case to infer NOT ATMOSPHERE-CONTACT. Again we have to go through an intermediary which states that something is in contact with the atmosphere if and only if its exterior is. Without going through all the steps we get:

```
SS-10 (LIQUID-IN PAINT-BRUSH1 PAINT1)      ;That the brush is in the paint
  LEADS-TO: (AND ((PAINTING7C SS-1)        ;satisfies the prerequsite
                 ((LIQUID-IN1 SS-10)))     ;LIQUID-IN1

SD-5 (NOT(ATMOSPHERE-CONTACT PAINT-BRUSH1);which allows us to infer
  COMES-FROM: ((LIQUID-IN4 SS-10))         ;no ATMOSPHERE-CONTACT for the
  LEADS-TO: ((ATMOSPHERE-CONTACT4 SD-5))   ;brush, which inturn implies

SD-6 (NOT(ATMOSPHERE-CONTACT SOME-PAINT1));the same for the paint.
  COMES-FROM: ((ATMOSPHERE-CONTACT5 SD-5));(The significance of this LEADS-
  LEADS-TO:  ((EVAP1 SD-7))                ;-TO will be explained shortly.)
```

SD-6 however directly contradicts SD-3, so we do contradiction read time inferences to clear up the problem. We first indicate that SD-3 is updated.

```
SD-3 (ATMOSPHERE-CONTACT SOME-PAINT1)
  COMES-FROM: ((ATMOSPHERE-CONTACT2 SD-3))
  LEADS-TO: ((EVAP1 SD-2))
  UPDATED-BY: ((SD-5))
```

However, SD-3 LEADS-TO further inferences, so they too must be updated. This will give us:

```
    SD-2 (EVAPORATION SOME-PAINT1)       ;Since we inferred EVAPORATION it must
      COMES-FROM: ((EVAP2 SD-2))         ;be updated.  To do this we add
      LEADS-TO: ((PAINT-DRY2 SD-1))      ;its negation, SD-7.  Also, we must
      UPDATED-BY: ((SD-7))               ;now follow the consequences of SD-2.

    SD-4 (NOT (ABSORBANT PAINT-BRUSH1)) ;which happens to be SD-4.
      COMES-FROM: ((PAINT-DRY4 SD-1))
      LEADS-TO: ((PAINTING-ABSORB SS-1))
      UPDATED-BY: ((SD-8))

   SD-7 (NOT (EVAPORATION SOME-PAINT1));SD-7 and 8 update SD-2 and 4
     COMES-FROM: ((EVAP2 SD-6))
     LEADS-TO: ((PAINT-DRY2 SD-9))      ;SD-9 is the negation of PAINT-DRY.
                                        ;Exactly how it is inferred is a compli-
   SD-7 (ABSORBANT PAINT-BRUSH1)        ;cation I would rather ignore.
     COMES-FROM: ((PAINT-DRY4 SD-8))
```

Note that the COMES-FROM link on SD-7 indicates that it was inferred via the rule which includes EVAP2. This is also the point of the LEADS-TO link on S-6. Together they state that since there is no longer atmosphere contact, there is no longer evaporation.

We now have corrected our false conclusion. So when the next line comes in, the question "Did the brush become unabsorbant?" there is no problem in answering "No".

7. MALAPROPISMS

Like virtually all experimental comprehension systems, Ms. Malaprop is a very delicate creature because of various problems. We will start will problems with the program proper.

7.1 Problems with the Program

Let us consider a few examples which Ms. Malaprop will screw up, and will screw up in ways one would never anticipate on the basis of the previous discussion.

> Jack was going to paint. He cleaned the brush.
>
> Q: Why? A: In order not to have paint on the brush.

This example is especially surprising since I claimed earlier that I did not want Ms. Malaprop to be fooled by "Jack was going to paint. He WASHED the brush." Indeed, Ms. Malaprop will work fine on this example, but the seemingly small difference between "clean" and "wash" is enough to cause havoc.

To understand what is happening we must first know that while "wash" is represented by the frame WASH, "clean" is represented by a statement of the form "cause an unmentioned fluid not to be STICKY-ON the thing cleaned". This is, to a first approximation, quite reasonable, at least given the representation assumptions which underlie the program. As far as I can tell there is no general knowledge of "cleaning" to warrant a separate frame, hence "clean" is not a predicate in its own right. "Wash" however is quite different. To perform this action we typically get a cleaning fluid STICKY-ON (and perphaps ABSORBED-BY) the object (put the clothes in the washer and let the water run in). Frequently if the cleaning fluid is water we will mix soap with it. Then we try to bring about homogenity between the cleaning fluid and the "bad stuff" on the object ... So the distinction between wash and clean is fine.

The trouble arises then when we try to integrate the "clean" statement.

(CAUSE JACK1 (NOT (STICKY-ON (OBJECT PAINT-BRUSH1)))).

CAUSE simply says to integrate NOT STICKY-ON. On the basis of the predicate there are three possibilities, the initial and final cleaning, and the getting paint on the brush in the center of PAINTING. We quickly eliminate the final clean on the basis of time, and the initial cleaning since PAINT-BRUSH1 is not something one normally paints. This leaves PAINTING4 (getting paint on the brush), and unfortunately there is a match, since PAINT-BRUSH1 is a very good INSTRUMENT, and our definition of "clean" leaves the fluid unmentioned, so the matcher figures it might, after all, be paint! We then link the NOT STICKY-ON statement with PAINTING4. Of course, it matches the negation of PAINTING4, but this only means that a frame expectation was confounded. Then when we ask the question ... Well, I am sure you can imagine the rest.

Part of the problem is one which came up earlier in the distinction between (NOT (CAUSE X)) and (CAUSE (NOT X)). At the moment Ms. Malaprop does not distinguish them. Note that if we had said "Jack was not able to get anything on the brush", we could reasonably interpret this as saying that PAINTING4 was not achieved. That is to say, we should count (NOT (CAUSE X)) as confounding our X pectations, but not (CAUSE (NOT X)). (Actually, things are more complicated. A better statement would be that (NOT (CAUSE X)) should be considered a better match than (CAUSE (NOT X)).) Another part of the problem is that whenever we say (CAUSE (NOT X)) there is an assumption that X was previously true. In particular "clean Y" suggests something is STICKY-ON Y. If we were able to express and use this fact it would also help deciding that PAINTING4 is not a good match since if the PAINT were already STICKY-ON INSTRUMENT, there would be no reason to invoke PAINTING4 in the first place.

Another problem with Ms. Malaprop is the lack of good facilities for handling time. Consider the following example:

> Jack was painting a chair. He dipped the brush into the paint.
>
> Some time later he finished. He washed the brush.
>
> Q: Was there paint on the brush before Jack washed it? A: No

There are several problems here, but they all revolve around the problem of time. The most immediate problem is that at the present time, when we learn that Jack washed the brush, Ms. Malaprop infers (NOT (STICKY-ON PAINT1 BRUSH1)) via a needed fact inference. This is fine, but troubles occur when Ms. Malaprop notes that this contradicts the (STICKY-ON PAINT1 BRUSH1) statement which was inferred (needed fact again) in line two. Ms. Malaprop assumes that the former line was never true, and that we simply made a mistake in the inference. This could be correct, providing we put in the machinery to see if the two statements are meant to be true at the same time. This could be done, although it will not be simple to get things so that this example would be considered an update, while the example of the last section, (ATMOSPHERE-CONTACT), would be considered a contradiction.

With this plus a few more corrections this example could be answered correctly. But only because the situation is a particularly simple one. Malaprop is missing one crucial thing: backup. To see why this is important, one only need consider situations where there is more than one occasion where paint was on the brush. Ms. Malaprop handles time by adding separate statements which state that some other statement was true at a certain time. Given this method of handleing time, when we look for a STICKY-ON statement we will first find one of them. We will then see if the time of this one is what we want, and if it isn't we then have to consider another of the occasions. It is this kind of situation, where we have no way of intelligently guiding the computer to finding the right answer first, that backup is useful. Its lack in Ms. Malaprop is a major reason why I have not bother to improve the time facilities.

But probably the major problem with Ms. Malaprop is that of search. This became particularly apparent when I tried switching to the domain of restaurants to see to what degree the programs design was influenced by the domain of painting. The results were mixed. However, for the most part the problems were with the program rather than the representation. (As should become apparent from the difficulities I mention, I have not run Ms. Malaprop on restaurant stories.) Of these, the two most crucial problems where time, which we have already noted, and search.

The search problem is, in essence, where shall we go when looking for frame statements which match the input. I have not discussed the issue much here, but as should be clear from the previous discussion, Ms. Malaprop only looks in currently active COMPLEX-EVENT frames. This worked for painting given a little fined tuning such as allowing matches against binders for frames pointed to by LEADS-TO or COMES-FROM pointers, e.g., (LIQUID-IN INSTRUMENT PAINT). With restaurants however, this will not work unless we are willing to put up with a very high degree of repeated, non-modular, information (which is, in fact, what SAM [11] does). For example, consider the story;

Jack went to a restaurant. The menu was in chinese.

Q: Did Jack have any trouble? A: He did not know what to order.

To handle this, we would start with something like the following piece of the restaurant frame.

```
    (KNOW AGENT MENU) COMES-FROM: ((READING3))
```

Plus we would need the following more general frames:

```
    (READING  (SIMPLE-EVENT)
       VARS: ...
      EVENT:  (AND READING1 (SEE READER READ)
                    READING2 (KNOW-LANGUAGE READER LANGUAGE-OF-READ))
                CAUSE
              READING3 (KNOW READER READ))

    (KNOW-LANGUAGE (STATE)
       VARS: ...
      EVENT: KNOW-L1 (KNOW-LANGUAGE AGENT LANGUAGE)
              IFF (SOMETIMES)
             KNOW-L2 (EQUAL LANGUAGE 'ENGLISH) )
```

(Needless to say, the specifics of READING and KNOW-LANGUAGE should not be taken seriously.) To handle the above example, Malaprop should make a needed fact read time inference to the effect that the agent will not know the menu. The problem is that the input does not match anything in RESTAURANT, but rather the negation of KNOW-L2, which is two levels below RESTAURANT. Unless we make it a standard practice to search subframes of complex-event frames or, as seems more reasonable, allow for more "bottom up" kinds of search this match, and the consequent needed fact inference, will never be made. (I might note that in a previous paper [16] I introduced the notion of a "restricted search" in order to handle some of the search problems in Ms. Malaprop. This was implemented, but has not proved general enough and in particular will not handle the above example.)

7.2 Problems with Knowledge

A second problem with the current program is that some of the knowledge encouded in the frames is not really correct. By this I do not mean that it is not scientifically correct, but rather that it does not corespond to our common sense understanding of the situation. Let me restrict myself to one such example. The reader may have noted some problems in our definition of PAINT-DRY, which we saw in section 2.3.

```
(PAINT-DRY (SIMPLE-EVENT)
   VARS: ...
  EVENT: (AND                                ;If there is paint sticking
            PAINT-DRY1 (STICKY-ON PAINT OBJECT) ;to an object, and it evap-
            PAINT-DRY2 (EVAPORATION PAINT) )    ;orates,
         CAUSES
         (AND                                ;then the paint will become
            PAINT-DRY3 (PART-OF PAINT OBJECT)   ;part of the object, and the
            PAINT-DRY4 (NOT (ABSORBENT OBJECT)));object becomes unabsorbent.
```

The first thing which might attract your attention is that everything except for EVAPORATION is a state. Why is there this exception. But as soon as one focues on this part of the rule, it should become clear that as it stands the rule is incorrect. The paint does not dry if there is any amount of evaporation, as the rule states, but only if the paint completely evaporates away.

A way around this problem, which would at the same time make things uniformaly states, would be to replace EVAPORATION which a statement which says, in effect, that the paint must be evaporated away (a state). The only reason I did not do this is that the best way to state this is by no means obvious, and the extra work would not show any imediate payoff. A first approximation would be to state that the volume of the paint should be reduced to zero. The problem here is that it ignores the volume of the paint residue. One might then say that paint consists of a liquid part and a solid part, and the condition is that the volume of the liquid part should be reduced to zero. Or should we say that the liquid part simply ceases to exist. Or should we veiw paint as simply a liquid, which, under the appropriate circumstances turns into a solid? In this case the the condition in PAINT-DRY should be that the paint is a solid. Furthermore, what are the conditions under which the liquid part goes away or the paint becomes a solid? Well it must be exposed long enough to the air and hence evaporate. But how long is long enough? Well, it depends on the volume of liquid and the circulation of air, and the temperature. How much on each? What are the constants of perportionality? But do people even know these things? I would doubt it, but how do we do without them?

Surely all of these questions have answers, and the answers may even be expressable within the frame representation given here. But there is a lot of work to be done.

7.3 Problems with the Frame Representation

There are some major gaps in the frame representation as it currently stands. Two of these have to do with "having" and "location". Consider:

> Jack had to do some painting. He did not have paint.
>
> Q: Could Jack do the painting?

This example cannot be represented for several reasons. For example Ms. Malaprop does not have any way to represent "had to" or "could". These however are tough problems for everybody. That I currently have no way to represent "not having" something is more unusual. To understand the problem we must realize that Ms. Malaprop handles "have" in an unusual, but I think reasonable way.

It has long been noted that the "have" relation "I have a pencil" is very different than in "I have a TV set". Typically this is understood as an ambiguity in "have". The problem (but I do not care to argue this point fully) is that under this view the number of possible "meanings" of "have" is extraordinarily large. The typical "solution" of "hold" vs. "own" is not sufficient, and to say that "have" can also mean "control" is to substitute one ambiguous word for another. Ms. Malaprop's approach is to treat "PERSON HAS THING" as usually indicating that THING serves as a binding for a variable in some complex event frame in which PERSON participates. (To flesh this out we will have to retain OWN, as well as allow HAVE to serve as an ambiguous but primitive predicate in the system so that it will be possible to learn only later the intended use.) In this way we can understand the particulars of the HAVE relation on the basis of what is required in the complex event frame, be it "hold" or "be in the proximity of" or whatever. Furthermore we can now do without the HAVE prerequisites which people are always putting in front of every action. These have always bothered me, since I never was sure what HAVE was suppose to mean in these cases (and usually it varied from case to case). Now all of these HAVES are replaced by a rather obvious convention on frames. To do an action in a frame, all of the variables mentioned in the action must be bound.

The trouble here is how do we represent "not have". Well, the obvious thing will be "some variable in some complex event frame cannot be bound". Note that simply saying that a variable is not bound is not sufficient, since Ms. Malaprop's normal convention when finding an unbound variable is to assume that the story teller has simply forgotten to mention the binding object. This is a good assumption, but it means that we need some way to say "this variable is unbindable, at least at the current time". So we can invent a predicate UNBINDABLE, but implementing the machinery to make use of it will be a non trivial task.

A similar problem to that of "have" is illustrated in the following example.

> Jack was painting. He dipped the brush in the paint.
>
> Q: Where is the brush.

As with "not have", Ms. Malaprop currently has no way to represent "where" questions, and again the cause is an peculiarity of the representation, though possibly a good one. Ms. Malaprop does not currently have the concept of a "location". One can assert various locative statements, such as LIQUID-IN, or CONTACT but there is no concept of AT, in the sense of (AT OBJECT LOCATION) where LOCATION is an entitiy describing the patch of space being taken up by OBJECT. This makes is difficult to compare various spacial predicates (as we would want to do in noting a change of position) and also makes it difficult to represent "where" questions. One way around this problem is to introduce AT, but utilize it as a pointer to the most specific information we have about location. Note the similarity here to the problem of ISA described in section 5.4 . There I suggested the need for a pointer to the most specific object information about a given object. Again, however, the status of these pointers is unclear at the current time.

So far we have looked at situations where the frame representation has made it difficult to express a particular concept. Now let us consider some more general problems. One difficulty which the reader has no doubt noted on his own, is that the use of COMES-FROM and LEADS-TO pointers to individual frame statements makes it quite difficult to write (or read) frames since they are constantly referring to the details of many other frames. Indeed, to some degree this defeats the modularity I seek. I think that with some clevar programming, this could be eliminated in PAINTING, by referring instead to the frame in which the statement appears and then making use of the matching constraint to insure that the correct target statement is located.

This may indeed prove to be the solution, but it is not obvious that it will work. Consider one of the schemes I was thinking about for use in the restraurant frames. The idea is that RESTAURANT will not be represented by a single frame. Instead there will be one frame (a COMPLEX-EVENT, naturally) which will include the knowledge of how one acts the part of the custormer. There will be a second for the waiter, and perphaps others. The frames (I only indicate two here) are then interconnected with LEADS-TO and COMES-FROM pointers.

```
RESTAURANTING (COMPLEX-EVENT)            WAITERING (COMPLEX-EVENT)
  VARS: ...                                VARS: ...
  EVENT: (IN AGENT RESTAURANT)             EVENT: (ON DISHES TABLE)
         (SEATED AGENT)                           (AT MENU CLIENT)
         (AT MENU AGENT)                          (AT WAITER CLIENT)

           COMES-FROM:                            (TELL CLIENT WAITER CHOICE)
         (AT SERVER AGENT)                          COMES-FROM:
           COMES-FROM:                            (TELL WAITER COOK CHOICE)
         (TELL AGENT SERVER CHOICE)               (GET-FROM WAITER FOOD COOK)

         (AT FOOD TABLE)                          (AT FOOD TABLE)
           COMES-FROM:                            (EAT CLIENT FOOD)
         (EAT AGENT FOOD)                           COMES-FROM:
         (AT BILL AGENT)                          (AT BILL CLIENT)
           COMES-FROM:                            (ON TIP TABLE)
         (ON TIP TABLE)                             COMES-FROM:
                                                  (PICK-UP WAITER TIP)

                                                  (NOT (ON DISHES TABLE))
```

This is very schematized, and the particulars are not to be taken seriously, but the general idea of separating the two processes seems quite plausible to me. However, given the complexity of the interaction, I have doubts that the LEADS-TO and COMES-FROM links can simply refer to, say, the WAITERING frame. It would seem that they would have to single out the particular line. On the other hand, perhaps such complex interaction between frames is a bad idea. At any rate it is something to think about.

Another problem with the representation is the use of intermediaries. They were introduced in order to allow a COMES-FROM or LEADS-TO pointer to something which was not a direct match, but which could be made to match by applying some rule. An unfortunate side effect has been to introduce a new, and unwanted degree of freedom in the construction of frames. For example, do we want the command to clean the instrument to lead to PAINT-DRY, or instead to lead to NOT ABSORBANT, using PAINT-DRY as an intermediary. The formalism says nothing about this. Instead the decision has been made on the ad hoc level of the results we will get when we ask "why should one wash the brush". Given my earlier comments about the need for a more sophisticated question answering section anyway, this is hardly good grounds for the decision.

One possible solution would be to eliminate intermediaries. However my current experience suggests that this is not practical. More reasonable would be to place a restriction of one intermediary per COMES-FROM or LEADS-TO pointer. One beneficial effect would be to eliminate cases where the use of intermediaries has gotten out of hand through the use of long chains of them. (Out of embarasment I have avoided showing any of these situations.) On the other hand, this restriction would not solve the problem mentioned in the last paragraph. That must wait for more and better restrictions on the formalism.

8 CONCLUSION

The last section discussed some of the problems with the program. Let me conclude on a more upbeat note.

Basically I see Ms. Malaprop in two ways. When viewed as a program, separate from the frame representation it uses, Ms. Malaprop is a working example of several ideas which are current in the AI literature: the frame recognition hypothesis, default values in the matching process, controlled read time inference, dependency relations to undo false conclusions, etc. But as I see it, the primary importance of the program is the frame representation it uses. The frame representation used here, while having its problems, is fairly successful in terms of the goals set out at the beginning of this paper.

Modularity. Probably the major force in the design of the frame representation is the goal of modularity. The restriction that complex events cannot contain causal relations was instituted solely to push the user into the difficult process of deciding what the basic cause and effect relations are, and expressing them in the most general form possible. Adjunct frames also facilitate modularity by allowing special ways of doing things to be separated from the general case, while still allowing the special cases to make use of the general information. This of course, is just inheritance of properties. What makes adjunct frames unusual is the degree to which they can modify the particulars of the master frame.

Worldly vs. control knowledge. The frame representation keeps a fairly clear separation between the facts it knows and information about how these facts are to be used in particular cases. Most of each frame consists of the basic facts. The knowledge of use does not strike the eye so readily, but it is also there. The most obvious place is the IF-NEEDED sections of the frames. The COMES-FROM and LEADS-TO pointers also specify control in the sense that they not only point out what other rule or event comes into play, but also specify whether this event is a reason (LEADS-TO), or if instead it is there to achieve a goal (COMES-FROM). In the latter case the pointer also indicates which result of the action is the one we are interested in.

Cleanliness. Throughout the description of the representation I have endevored to keep the representation clean by specifying exactly what may and may not go into the representation. COMES-FROM and LEADS-TO pointers must link matching statements, and if they don't then intermediaries must be given which specify how a match is to be brought about. Complex event frames may not directly specify cause and effect relations, while simple events may only specify one. A complex event may only indicate what we wish to happen, and not anything which might happen at the same time, such as mistakes. The use of arbitrary LISP programs has been severely reduced. As indicated in section 5, all but one of the occasions where they currently appear will be eliminated by further refinements of the program, either improved question answering facilities, or by making into representation primitives those which seem inextricably LISPish (e.g., EQUAL). Finally, the restriction that all predicates are frames has pushed the semantic representation into a few unusual, but I think interesting directions; most noticeably the elimination of ISA, HAVE, and AT. Admittedly I noted some problems related to their absence, but these problems are ones which previous systems papered over by the use of predicates whoes meaning was unclear.

Problem solving. Early on we noted that the representation of PAINTING was wholly in terms of states to be acheived, rather than actions to be performed. The idea here is that since the basic logic of PAINTING, as well as most other goal directed activities is in terms of goal states and subgoal states, it would be easier to connect our knowedge of PAINTING to problem solving programs if PAINTING were itself represented in that form. Furthermore, by keeping track of why things are done (via the COMES-FROM and LEADS-TO links) we have the information necessary to handle unusual cases. Finally, because of the enforced modularity, we not only have available the standard ways of doing things, but the more basic cause and effect relations which would be needed should the standard ways break down. Of course, Ms. Malaprop does not have the mechanisms to use this information in problem solving, such as proper indexing of ways to do things, or subgoal protection. It is an open question whether such things are even needed for story comprehension. But what does seem clear is that the basic strata of common sense information can be represented so that it can be used in both activities.

ACKNOWLEDGEMENTS

The research for, and the early drafts of, the program described here were done while I was at the Institute for Semantic and Cognitive Studies at the University of Geneva, sponsored by the university and the Dalle Molle Foundation. During this period the members of the institute were very helpful in criticizing proposed solutions, and providing alternatives, in particular Maggie King and Walter Bischof. The implementation of the program, and the writing of this paper were done at the Yale Artificial Intelligence Project.

Here I benfitied from conversations with many people, in particular Drew McDermott. I would also like to thank Dave Barstow, Larry Birnbaum, Jerry DeJong, and Drew McDermott who read and extensively commented an earlier version of this paper. The Yale A.I. Project is funded in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-75-C-1111.

REFERENCES

1. Minsky, M., A framework for representing knowledge, in: P. H. Winston, ed., _The Psychology of Computer Vision_ (McGraw-Hill, New York, 1975).
2. Schank, R., and Abelson, R., _Scripts, Plans, Goals and Understanding_ (Lawrence Erlbaum Associates, Inc. Hillsdale, New Jersey, 1977).
3. Bobrow, D. G., and Winograd, T., A knowledge representation language, _Journal of Cognitive Science_, 1, 1, (January 1977) 1-46.
4. Hayes, P. J., Some association-based techniques for lexical disambiguation by machine, TR25, Computer Science Department, The University of Rochester (June 1977).
5. Rieger, C., The common sense algorithm as a basis for computer models of human memory, in: R. Schank and B. L. Nash-Webber eds., _Proceedings of the Conference on Theoretical Issues in Natural Language Processing,_ (1975).
6. Wilks, Y., Natural language inference, AIM-211, Stanford Artificial Intelligence Laboratory (August, 1973).
7. Charniak, E., A framed PAINTING: the representation of a common sense knowledge fragment, _Journal of Cognitive Science_, 1, 4, (October 1977)

355-395.

8. Hewitt, C., Description and theoretical analysis (using schemata) of PLANNER, TR-258, M.I.T. A.I. Lab, 1972.

9. Kowalski, R., Predicate logic as programming language, Memo 70, University of Edinburgh Department of Computational Logic, 1973.

10. Charniak, E., Organization and inference in a frame like system of common sense knowledge, in: R. Schank and B. L. Nash-Webber eds., Proceedings of the Conference on Theoretical Issues in Natural Language Processing, (1975).

11. Cullingford, R., Script application: computer understanding of newspaper stories, Research Report 116, Yale University Department of Computer Science, (January 1978).

12. Charniak, E., Inference and knowlege 1, in: E. Charniak and Y. Wilks eds. Computational Semantics, (North Holland, Amsterdam, 1976).

13. Doyle, J., Truth maintenance systems for problem solving, TR-419, M.I.T. A.I. Lab, (1977).

14. McDermott, D. Flexibility and efficiency in a computer program for designing circuits, TR-402, M.I.T. A.I. Lab, (June 1977).

15. Lehnert, W., The process of question answering, Research Report 88, Yale University Department of Computer Science, (May 1977).

16. Charniak, E., Ms. Malaprop, a language comprehension program, in: Proceedings of the 5th International Joint Conference on Artificial Intelligence, (1977).

APPENDIX

```
(PAINTING (COMPLEX-EVENT)
   VARS: (AGENT (ANIMATE AGENT))
         (OBJECT (SOLID OBJECT))
         (PAINT (LIQUID PAINT)
           NORMAL: (PAINT PAINT))
         (SOME-PAINT (AND (LIQUID SOME-PAINT)
                          (PART-OF SOME-PAINT PAINT)))
         (SOME-PAINT-VOL (VOLUME SOME-PAINT-VOL SOME-PAINT))
         (INSTRUMENT (SOLID INSTRUMENT)
           NORMAL: PAINTING-BRUSH (PAINT-BRUSH INSTRUMENT)
                   PAINTING-ROLLER (ROLLER INSTRUMENT)
                   PAINTING-ABSORB (ABSORBANT INSTRUMENT))
         (DIRT (DIRT DIRT))
         (PAPER (SOLID PAPER) NORMAL: (NEWSPAPER PAPER))
   GOAL: PAINTING-GOAL (EXTERIOR PAINT OBJECT)
           COMES-FROM: ((via intermediaries) (PAINTING5))
  EVENT: PAINTING1 (NOT (STICKY-ON DIRT OBJECT))
           COMES-FROM: ((WASH-GOAL))
           LEADS-TO:  ((FLAKING1))
         PAINTING2 (UNDER PAPER OBJECT)
         PAINTING3
          (LOOP PAINTING4 (STICKY-ON PAINT INSTRUMENT)
                 COMES-FROM: ((LIQUID-IN2 PAINTING4C
                                          (LIQUID-IN INSTRUMENT PAINT)))
                PAINTING5 (CONTACT INSTRUMENT OBJECT)
                PAINTING6 (THRESHHOLD-UNDER SOME-PAINT-VOL)
                 LEADS-TO: ((DRIP1))
                  ;Make sure there is not too much paint on the instrument.
           )
         PAINTING7 (NOT (ATMOSPHERE-CONTACT SOME-PAINT))
          ;Until the instrument is cleaned, keep the paint on it out of the air.
           COMES-FROM: ((ATMOSPHERE-CONTACT5 ATMOSPHERE-CONTACT4)
                        ;If we keep INSTRUMENT out, then anything on it will
                        ;be out of the air also,
                       (LIQUID-IN4 PAINTING7C (LIQUID-IN INSTRUMENT PAINT)))

                        ;so keep INSTRUMENT in the paint.
        PAINTING8 (NOT (STICKY-ON SOME-PAINT INSTRUMENT))
          COMES-FROM: (AND ((WIPE-GOAL PAINTING8C (WIPE INSTRUMENT PAPER)))
                       ;Clean INSTRUMENT by wipeing and washing.
                         ((WASH-GOAL)))
          LEADS-TO: ((PAINT-DRY1 PAINT-DRY4)(PAINTING-ABSORB))     )
```